Tracker Program – User Guide

Ed Kozak
Serenica Technologies Corporation

Tom Gee
Covenant Systems Development

Serenica Technologies Corporation
250 Florence Ave, Toronto, Ontario, M2N 1G6

Project Manager: Brad Cain 416-635-2195

Contract Number: W7711-037866/001/TOR

Contract Scientific Authority: Brad Cain 416-635-2195

Defence Research & Development Canada - Toronto

Contractor Report

CR 2004-027

March 2004

105-05812/

Author

Ed Kozak

Approved by

Brad Cain
Scientific Authority

Approved for release by

Brad Cain
Scientific Authority

The scientific or technical validity of this Contractor Report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

© Her Majesty the Queen as represented by the Minister of National Defence, 2004

© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2004

Table of contents

Table of contents i

Abstract iv

Résumé iv

Executive summary v

Sommaire vii

1 Introduction 1

2 Description 1

2.1 Using Tracker 2

2.1.1 Installing 2

2.1.1.1 On Linux 2

2.1.1.2 On Windows 2

2.1.2 Configuring 3

2.1.2.1 Creating a Hardware Configuration File . . 3

2.1.2.2 Creating a Subject Information File 7

2.1.2.3 Creating a Trial Configuration File 8

2.1.3 Running 8

3 Configuration Parameters and Files 9

3.1 Equipment 9

3.1.1 Screen 10

3.1.2 Clock 10

3.1.3 Mouse 10

3.1.4 Joystick 11

3.1.5 XML Format for Equipment 12

3.2 Subject . . . 13
3.2.1 Text Format . . . 13
3.2.2 XML Format for Subject . . . 13
3.3 Trial . . . 13
3.3.1 General Settings . . . 14
3.3.2 Graphics . . . 15
3.3.3 Window Specification . . . 15
3.3.4 Target and Cursor Specification . . . 16
3.3.5 User Input . . . 17
3.3.5.1 Joystick . . . 18
3.3.5.2 Mouse . . . 19
3.3.5.3 Keyboard . . . 20
3.3.6 Kinematics . . . 21
3.3.6.1 Movement and Mass of Axes . . . 21
3.3.6.2 Motion Epsilons . . . 22
3.3.6.3 Motion Equations . . . 23
3.3.7 XML Format for Trial . . . 24
Annexes . . . 27
A Example Files . . . 27
A.1 Equipment Configuration . . . 27
A.2 Subject Configuration . . . 27
A.3 Trial Configuration . . . 27
A.4 Output Data File . . . 31
B Recommended Platform . . . 32

C Software Requirements and Procedures 33

References 35

Index 36

Abstract

This document is a user guide for and description of a computer-based, manual tracking task. The tracking task was developed to support experiments and modelling of human performance at DRDC Toronto. To this end, the program allows greater data sampling and collection rates than typically reported by other investigations within the literature; the acquisition and recording rates will depend upon the computer hardware and software used to run the tracking task program. The tracking task program is designed for use on Intel-based computers running Linux Redhat Fedora as well as Microsoft Windows NT operating systems. In order to work on Windows environments, additional, freely-available but non-standard software must be added such as Cygwin, mxml and SDL to execute; few modifications should be required for use on other Linux operating systems. The program provides pursuit, compensatory and general (pursuit and compensatory) tracking tasks. The experimenter can create a forcing function from a linear combination of any number of sinusoids or step inputs.

Résumé

Le présent document est le guide d'utilisateur et la description d'une tâche de poursuite manuelle informatisée. Cette tâche de poursuite a été mise au point pour soutenir les expériences et la modélisation de la performance humaine à RDDC Toronto. À cette fin, le programme permet de plus grandes fréquences d'échantillonnage et de collecte de données que celles qu'on observe généralement dans la documentation d'autres investigations ; les taux d'acquisition et d'enregistrement dépendront du matériel et du logiciel utilisés pour exécuter le programme de tâches de poursuite. Le programme de tâches de poursuite est conçu pour des ordinateurs Intel tournant sous les systèmes d'exploitation Linux Redhat Fedora ainsi que Microsoft Windows NT. Afin de l'exécuter dans des environnements Windows, il faut ajouter des logiciels supplémentaires, gratuits mais non standard, comme Cygwin, mxml et SDL ; peu de modifications devraient être nécessaires pour une utilisation sous d'autres systèmes d'exploitation Linux. Le programme offre des tâches de poursuite mobiles, compensatrices et générales (mobiles et compensatrices). L'expérimentateur peut créer une fonction de forçage à partir d'une combinaison linéaire de n'importe quel nombre de sinusoïdes ou de signaux d'entrée progressifs.

Executive summary

This document is a user guide for and description of a computer-based, manual tracking task. The tracking task was developed to support experiments and modelling of human performance at DRDC Toronto. To this end, the program allows greater data sampling and collection rates than typically reported by other investigations within the literature; the acquisition and recording rates will depend upon the computer hardware and software used to run the tracking task program. Only executable versions tracking task programs were found in an informal, web-based search, so modification of other programs to meet DRDC needs would not be practicable. This program is a tool for researchers wishing to study human performance in continuous tasks in a laboratory setting and could be extended to include other tasks, allowing researchers to explore workload and decision-making strategies in multi-task environments.

The tracking task program is designed for use on Intel-based computers running Linux Redhat Fedora as well as Microsoft Windows NT operating systems. In order to work on Windows environments, additional, freely-available but non-standard software must be added such as Cygwin, mxml and SDL to execute; few modifications should be required for use on other Linux operating systems. At this point, the program provides the basic capabilities required for currently-planned studies in the SMART section of DRDC Toronto. The software has been bench-marked on a 1 GHz Pentium III laptop and found to meet the 100 Hz sample rate requirement and can make use of the mouse, joystick or keyboard keys to control the cursor position as desired.

The program provides pursuit, compensatory and general (pursuit and compensatory) tracking tasks. The experimenter can create a forcing function from a linear combination of any number of sinusoids or step inputs. Coefficients of the sinusoidal functions (amplitude, frequency, phase, and offset) may be specified for each term; coefficients of the step inputs (amplitude, time of occurrence, and frequency of occurrence) may be similarly specified. Other forcing functions may be added by the experimenter as required. The forcing functions can currently be specified to represent position, velocity, or acceleration (zero, first, or second order) control systems.

Currently, a single shape, a chevron, has been defined for the cursor and target, although the attributes of the chevron (size, shape, colour) may be specified independently for each of the cursor and target. The experimenter may develop other shapes as desired.

The program provides tools to calibrate the computer hardware, including the pointing devices, display characteristics, and data recording rates. Tools are also avail-

able to define experimental trial conditions and subject profiles. Alternatively, these configuration files may be created manually with a text editor or spreadsheet program. The configuration files are converted to XML input files for use by the tracking task program.

The source code is available if further development is required or if it is desired to integrate other tasks to look at multi-task operator performance; for instance, the user interface is command line and the program would benefit from a graphical user interface if it is found to be of widespread value. While the Department of National Defence owns the intellectual property rights to this program, the intent is that the program be made available to other researchers in the field, both within and outside of the Department of National Defence, with the provision that enhancements or corrections to the source code be made available for inclusion in newer versions.

Ed Kozak, Tom Gee; 2004; Tracker Program – User Guide; CR 2004-027; Defence Research & Development Canada - Toronto.

Sommaire

Le présent document constitue le guide d'utilisateur et la description d'une tâche de poursuite manuelle informatisée. Cette tâche de poursuite a été mise au point pour soutenir les expériences et la modélisation de la performance humaine à RDDC Toronto. À cette fin, le programme permet de plus grandes fréquences d'échantillonnage et de collecte de données que celles qu'on observe généralement dans la documentation d'autres investigations ; les taux d'acquisition et d'enregistrement dépendront du matériel et du logiciel utilisés pour exécuter le programme de tâches de poursuite. Une recherche simple sur le Web n'ayant permis de trouver que des programmes de tâche de poursuite de versions exécutables, la modification d'autres programmes pour répondre aux besoins de RDDC ne serait pas réalisable. Le présent programme est un outil destiné aux chercheurs qui veulent étudier la performance humaine dans des tâches continues en laboratoire et pourrait être étendu à d'autres tâches, afin de permettre l'exploration de stratégies de charge de travail et de prise de décision dans des environnements multitâches.

Le programme de tâches de poursuite est conçu pour des ordinateurs Intel tournant sous les systèmes d'exploitation Linux Redhat Fedora ainsi que Microsoft Windows NT. Afin de l'exécuter dans des environnements Windows, il faut ajouter des logiciels supplémentaires, gratuits mais non standard, comme Cygwin, mxml et SDL ; peu de modifications devraient être nécessaires pour une utilisation sous d'autres systèmes d'exploitation Linux. À ce stade, le programme offre les fonctions de base requises pour les études actuellement planifiées à la section SMART de RDDC Toronto. Le logiciel a été mis au banc d'essai sur un ordinateur portatif Pentium III 1 GHz pour répondre à une exigence de fréquence d'échantillonnage de 100 Hz et il permet de contrôler la position du curseur à volonté, à l'aide de la souris, du manche à balai ou des touches du clavier.

Le programme offre des tâches de poursuite mobiles, compensatrices et générales (mobiles et compensatrices). L'expérimentateur peut créer une fonction de forçage à partir d'une combinaison linéaire de n'importe quel nombre de sinusoïdes ou de signaux d'entrée progressifs. Les coefficients des fonctions sinusoïdales (amplitude, fréquence, phase et écart) peuvent être spécifiés pour chaque terme ; les coefficients des signaux d'entrée progressifs (amplitude, délai d'apparition et fréquence) peuvent être spécifiés de la même manière. L'expérimentateur peut ajouter d'autres fonctions de forçage au besoin. Les fonctions de forçage peuvent être spécifiées au fur et à mesure pour représenter des systèmes de contrôle de la position, de la vélocité ou de l'accélération (ordre zéro, premier ordre, second ordre).

Actuellement, une forme unique, le chevron, est défini pour le curseur et la cible, bien que les attributs du chevron (taille, forme, couleur) puissent être spécifiés séparément pour le curseur et pour la cible. L'expérimentateur peut développer

d'autres formes à son gré.

Le programme fournit des outils pour calibrer le matériel informatique, y compris les dispositifs de pointage, les caractéristiques d'affichage et les taux d'enregistrement des données. Il existe aussi des outils pour définir les conditions d'essai expérimental et les profils de sujet. On peut aussi créer ces fichiers de configuration manuellement au moyen d'un éditeur de texte ou d'un tableur. Les fichiers de configuration sont convertis en fichiers d'entrée XML utilisables par le programme de tâches de poursuite.

Le code source est accessible s'il faut développer davantage le programme ou si on veut intégrer d'autres tâches afin d'examiner la performance d'un opérateur multitâche ; par exemple, l'interface utilisateur est la ligne de commande et le programme aurait avantage à posséder une interface graphique si son utilisation devenait très répandue. Bien que le ministère de la Défense nationale soit le détenteur des droits de propriété intellectuelle de ce programme, le but visé est de rendre le programme accessible à d'autres chercheurs du domaine, à la fois à l'intérieur et à l'extérieur du ministère de la Défense nationale, à la condition que les améliorations ou les corrections apportées au code source soient fournies afin d'être intégrées aux versions ultérieures.

Ed Kozak, Tom Gee; 2004; Tracker Program – User Guide; CR 2004-027; Recherche et dévelopment pour la défense Canada - Toronto.

1 Introduction

The tracker program is designed to provide the experimenter with detailed control over a multiple order, multiple input visual tracking task. The experimenter can control the types of inputs, their application and the kinematics of a target and cursor with fine resolution.

Written in ANSI C++ and the open-source packages OpenGL [1], Simple Direct-Media Layer (SDL) [2], and mXML [3], the tracker program should be compatible with many different operating systems. The program will be tested on Linux and Microsoft Windows NT and 2000, but should also run on any machine supported by SDL and OpenGL, including MacOS, OS X and QNX.

2 Description

The tracker program consists of a single executable combined with a number of library files to be installed on a computer running Linux or Microsoft Windows. Accompanying the main executable is an input file conversion program for converting fixed-format files to XML [4] for use with the tracker program.

The tracker is invoked as a command-line program where the user specifies the output file and the input files.

```
> tracker out021.csv eqpt.xml subj021.xml trial001.xml
```

The input files typically consist of three configuration files where the first provides the parameters of the equipment, the second gives the subject information, and the third defines the specific parameters for the trial. The input files are internally concatenated before processing, so the user may divide up the configuration information in any number of ways, using a single configuration file or dozens of files. File names may be specified as desired by the experimenter to distinguish configuration files for various investigations.

The input file conversion program is capable of processing the configuration data according to the three part format described above, and uses a command line switch to determine which mode to run in.

```
> trackerxml --equipment eqpt1.cvs eqpt1.xml
> trackerxml --subject subj15.csv subj15.xml
> trackerxml --trial trial002.csv trial002.xml
```

2.1 Using Tracker

This section examines how the tracker program is typically used. The technical details are left to the later sections.

2.1.1 Installing

The tracker program can run under various flavours of Linux or Microsoft Windows. Although we could not provide a simple `setup` program, the installation is not difficult.

All of the files required are provided on the CD.

2.1.1.1 On Linux

Installation on linux is simply a matter of installing the pre-compiled rpm package. The rpm provided on the disk is for Fedora Linux. The binary rpm has dependencies on SDL and libGL, which are identified by the rpm itself, and can be satisfied by installing the appropriate rpm packages from the Linux distribution CDs. In the case of Fedora, these are the SDL and XFree86-Mesa-libGL packages.

If the precompiled binary rpm cannot be used, then the alternative is to build a binary rpm from the source rpm, provided on the CD. The source rpm has dependencies on nana and mxml packages, which can be found on the CD, and the SDL-devel and XFree86-devel packages, which can be obtained from the Linux distribution CD. To build the binary rpm, execute the following command:

```
> rpmbuild --rebuild tracker-1.0.0-1.src.rpm
```

On a Fedora/Redhat system, the resulting binary rpm package will be found in the directory `/usr/src/redhat/RPMS/i386`.

2.1.1.2 On Windows

To execute this program under Windows requires the installation of ActiveState Perl (ActivePerl) for the XML conversion tool and a few cygwin-compiled libraries for the real-time executable.

ActiveState uses their own external installer program of which there are two versions: `winNTinstaller.exe` for NT-based systems and `win95installer.exe` for non-NT versions of Windows. The first step is to run one of these two installers. This will install the ActiveState installer program on the computer.

Next, you must install ActivePerl. Since the ActiveState installer is now loaded, simply double-click on the ActivePerl icon. This completes the installation of ActivePerl.

The final step is load the remaining executables and DLLs onto the machine and ensure that the directories containing these files is part of the system PATH list. The simplest way to do this to place all of these files into a common directory and add that directory to the PATH list. The files are `SDL.dll`, `cygwin1.dll`, `tracker.exe`, `calibrate.exe` and `trackerxml.pl`.

So, for example, assuming your main disk is C and your CD is D, after installing `winXXinstaller` and ActivePerl, you would do the following:

```
$ cd C:
$ mkdir tracker
$ copy D:\SDL.dll tracker
$ copy D:\cygwin1.dll tracker
$ copy D:\tracker.exe tracker
$ copy D:\calibrate.exe tracker
$ copy D:\trackerxml.pl tracker
$ edit C:\autoexec.bat
```

Within the `autoexec.bat` file you would have an entry like:

```
SET PATH=C:\tracker;%PATH%
```

That completes the installation of tracker. To execute, simply invoke `tracker` with appropriate parameters via the Run menu item or from a batch file or MS-DOS prompt.

2.1.2 Configuring

To configure tracker, you must create three types configuration files: one type for the equipment, one for the subject and one for the trial. Typically you will have one equipment file describing the hardware you are using, one subject file for every subject in the study, and one trial configuration for every distinct trial in the study.

2.1.2.1 Creating a Hardware Configuration File

To determine the video resolution, screen dimensions, screen update rate, joystick limits, and so on, use the program `calibrate`. It expects a single argument which

is the output file name you wish to use. The hardware configuration file is typically created once, when the tracking program is installed. The hardware configuration file can be recreated at any time, if for example, new hardware is added to the system, or to compensate for hardware performance drift. If the computer hardware, say adding a new joystick, then it can be calibrated without repeating the other calibrations; the calibration program will reuse existing or default values if the user elects not to calibrate all hardware.

Under Linux, use

```
bash$ ./calibrate equipment.csv
```

under Windows:

```
> calibrate.exe equipment.csv
```

Calibration requires several steps. The following is the output of a session with the calibration program that we'll explain as we go:

```
(1) determining available resolutions

The screen will flash and flicker for the next
several seconds.

Ready to perform video calibration? [Y/n]
```

The uppercase letter (in this case `Y`) is the default behaviour. At each prompt, if in doubt, just hit 'return' to accept the default.

Running this test on a Dell laptop produced the following:

```
Available modes are:
1600 x 1200   5.4 Hz  (0)
1400 x 1050   0.6 Hz  (1)
1024 x 768    1.1 Hz  (2)
800 x 600     23.4 Hz (3)
640 x 480     27.6 Hz (4)

Desired mode to use? [0] 3
```

The first pair of numbers (e.g. 1600 x 1200) is one mode supported by your video card. The second number (e.g. 5.4 Hz) is estimated video frame rate the program will use at that resolution. The number in parentheses (e.g. 0) is the i.d. for that mode.

So, in this example, mode 0 was the default but I chose mode 3 (800 x 600, 23.4 Hz) for the higher update rate.

Once selected, `calibrate` responds with:

```
Good.  Using 800 x 600 at 23.4 Hz.


(2) determining timer resolution
Do you wish to calibrate timer resolution? [Y/n]
```

Hitting "return" selects the default, which is to calibrate the delay timer on this platform. The result is:

```
resolution = 10 ms
Does that sound reasonable? [Y/n]
```

The calibration of the timer uses a statistical process that sometimes fails. If the number looks unreasonable (e.g. more than 10 ms), you can answer 'n' to this question and the program will attempt the estimate again.

```
(3) calibrating display refresh rate

Do you wish to calibrate the refresh rate? [Y/n]
```

I selected the default which generates a more refined estimate of the update rate (originally set at 23.4 Hz in step 1). This value, like the others, is stored in the output file.

Unfortunately, the ability to detect refresh rate is dependent on the specific video chipset being used. At the moment, this computed value is only reliable for video chips made by nVidia. For all other systems, the value must be manually corrected. The correct value to enter is the vertical refresh rate of your monitor, which can be obtained from the monitor's manual or onboard adjustment menu.

```
(4) calibrating joystick axes
Do you wish to calibrate the joystick? [Y/n]

Please cycles all axes on the joystick to their extreme positions
several times.

Press button 0 on the joystick when this is complete.
32/32 32/32 0/32 0/ 0 0/32
```

The display shows the current limits of each axis of the joystick. When I press button zero, the program says:

```
Please center the joystick.

Press button 0 on the joystick when centered.
```

I do so, and press button zero again.

```
Press button 0 on the joystick again to end sampling.
```

Which completes the sampling of the centered section values.

This joystick calibration process typically does not need to be done for digital joysticks. Digital sticks should always read between -32767 and +32767 with centered value at 0. When using a digital stick, the calibration entry should be:

```
joystick calibration horizontal 32767.0 0.0 0.1
joystick calibration vertical   32767.0 0.0 0.1
```

We *highly* recommend using digital joysticks. They are inexpensive, do not drift and provide dependable, high-resolution, noise-free inputs.

```
(5) calibrating mouse axes
Do you wish to calibrate the mouse? [y/N]
```

Calibrating the mouse is almost never useful, so it defaults to 'N'.

```
(6) calibrating screen dimensions
Do you wish to calibrate screen dimensions? [Y/n]
```

I press "return" to accept the default yes.

```
The screen will go blank on this next run.
Two rulers, one horizontal and one vertical, will appear.
Please a (physical) ruler up against each ruler on the screen
and adjust the screen rulers lengths by using the arrow keys.
Pressing shift with the arrow keys will adjust the length
tenfold for rapid, course changes.

When hash marks on the screen rulers match the physical rulers
press the spacebar.

Ready to begin? [Y/n]
```

Once I've matched the displayed rulers against a physical ruler, I press the spacebar and the program is complete.

We have now generated an `equipment.csv` file which, on my laptop, looks something like this:

```
screen      calibration    horizontal     resolution     800
screen      calibration    vertical       resolution     600
clock calibration    resolution     10
screen      calibration    vertical       refresh 50.829372
joystick    calibration    axis 0  32767.0 0.0    0.1
joystick    calibration    axis 1  32767.0 0.0    0.1
screen      calibration    horizontal     distance       30.2
screen      calibration    vertical       distance       22.7
```

2.1.2.2 Creating a Subject Information File

A unique information file is created subject in an experiment, although, this file may be reused for other tracking experiments the subject participates in. This file contains descriptive information only that is used only to document the output file.

Use `samples/subject.csv` on the CD as an example. The only restriction on the format of the subject information is that it should not contain `<subject>` or `</subject>`, as these are the delimiters used by the XML parser. It should also not contain 8-bit text. So, basically anything you can enter in a word processor or spreadsheet is permitted.

2.1.2.3 Creating a Trial Configuration File

The trial configure file contains a unique set of values describing an experimental condition; several trial files may be generated to define a complete experiment. These trial configuration files are only created once, then may be used for any subject in an experiment, or even reused in other, similar experiments.

Use `samples/trial.csv` as an example. The details of the configuration options can be found in the Trial configuration section (3.3).

2.1.3 Running

1. After installing and creating appropriate equipment, subject and trial configuration files, you must convert the files into XML format for use by the tracker. If the output XML file already exists, use the argument `--force` to overwrite.

```
> trackerxml.pl --equipment equipment.csv equipment.xml
> trackerxml.pl --subject   subject.csv   subject.xml
> trackerxml.pl --trial     trial.csv     trial.xml
```

2. Run the tracker program by moving into the directory it was installed into (if in Windows) and invoking the executable.

```
> cd C:/Tracker
> tracker.exe output.csv *.xml
```

   or, under Linux

```
$ tracker output.csv *.xml
```

The output of the program will be stored in the file `output.csv`. If this file already exists, it will refuse to overwrite it and will simply stop running with a warning message.

The output file can then be loaded into any spreadsheet or other program capable of handling single-character-delimited files.

The tracker program will also create a file called `tracker.log`. This small file is used for debugging and tracing the basic behaviour of the system, and will be silently overwritten the next time `tracker` is run. If there is a problem with a specific run, please save this file before contacting Serenica.

If you wish to present the subject with multiple runs of different configurations, combine the `tracker.exe` calls into a single batch file or shell script with the different configuration and output files. For example, if you have three distinct trial

conditions configured in the files `trial00.xml`, `trial01.xml` and `trial02.xml`, create a batch script (say `subject01.bat`) and put the invocations into that file.

The resulting batch script might look like this under Windows:

```
tracker.exe output00.csv equipment.xml subject.xml trial00.xml
tracker.exe output01.csv equipment.xml subject.xml trial01.xml
tracker.exe output02.csv equipment.xml subject.xml trial02.xml
```

or under Linux:

```
tracker output00.csv equipment.xml subject.xml trial00.xml
tracker output01.csv equipment.xml subject.xml trial01.xml
tracker output02.csv equipment.xml subject.xml trial02.xml
```

3 Configuration Parameters and Files

We will describe the configuration files in terms of the three part division of equipment, subject and trial although as noted in the introduction to this section, such a division is required only by the conversion program and not by the tracker program itself.

This section documents the parameters and their meanings and units. The input file formats for the XML conversion program are documented, as are the XML formats used by the tracker program.

All input files for the conversion program are single-character delimited text files. By default, the delimiter is a tab although this may be changed using a command line argument. Any value in double-quotes ("such as this") is taken as a comment and ignored. This format is easily created using a spreadsheet or other supporting program such as Matlab. Blank rows are also allowed.

When we describe the input format, we are speaking of the text files input to the `trackerxml` conversion program to be reformatted into XML. The actual XML is given in a separate subsection at the end of each section.

3.1 Equipment

To simplify the description, we configure the screen, clock, mouse and joystick separately. A sample equipment configuration is given in annex A.1.

The equipment configuration describes the physical and electronic configuration of the hardware used for the experimental trial. The functions of the equipment – how they relate to the trial itself – are defined within the trial configuration file.

3.1.1 Screen

Entity	Function	Direction	Measure	Value
screen	calibration	horizontal	distance	*value*
screen	calibration	vertical	distance	*value*
screen	calibration	horizontal	resolution	*value*
screen	calibration	vertical	resolution	*value*
screen	calibration	vertical	refresh	*value*

Parameter	Units	Range	Default	Description
horizontal distance	cm	> 0	*none*	Physical width of the drawable area of the screen.
vertical distance	cm	> 0	*none*	Physical height of the drawable area of the screen.
horizontal resolution	pixels	> 0	*none*	Number of pixels horizontally on the screen.
vertical resolution	pixels	> 0	*none*	Number of pixels vertically on the screen.
vertical refresh	Hz	> 0	60	Number of frames displayed on the monitor per second.

3.1.2 Clock

Entity	Function	Measure	Value
clock	calibration	resolution	*value*

Parameter	Units	Range	Default	Description
resolution	ms	> 0	10 ms	Smallest interval of time within which the tick timer increments.

3.1.3 Mouse

Entity	Function	Direction	Slope	Offset
mouse	calibration	horizontal	*value*	*value*
mouse	calibration	vertical	*value*	*value*

Parameter	Units	Range	Default	Description
horizontal slope	count/cm	> 0	1	Rate of counts (Δx) per centimeter for horizontal mouse movement.
horizontal offset	count	> 0	0	Number of counts reported for horizontal position when the mouse is at the origin.
vertical slope	count/cm	> 0	1	Rate of counts (Δy) per centimeter for vertical mouse movement.
vertical offset	count	> 0	0	Number of counts reported for vertical position when the mouse is at the origin.

Slope and offset are used to map the reported mouse displacement (in counts) to a physical mouse displacement (in centimetres), as shown in equation 1. Because the mouse can be recentered by the user, the `offset` value is unlikely to be very useful.

$$\text{displacement}_{cm} = \frac{\text{displacement}_{counts} - \text{offset}}{\text{slope}} \tag{1}$$

3.1.4 Joystick

Entity	Function	Direction	Slope	Offset	Threshold
`joystick`	`calibration`	`horizontal`	*value*	*value*	*value*
`joystick`	`calibration`	`vertical`	*value*	*value*	*value*
`joystick`	`calibration`	`axis` *N*	*value*	*value*	*value*

Parameter	Units	Range	Default	Description
horizontal slope	count/%	> 0	1	Rate of counts per percentage of displacement measured from the far left (0.0) to the far right (1.0).
horizontal offset	count	> 0	0	Number of counts reported for horizontal position when the joystick is at the far left.
horizontal threshold	%	$\geq 0, \leq 1$	0.1	Horizontal "deadspace" around joystick center.
vertical slope	count/%	> 0	1	Rate of counts per percentage of displacement measured from the bottom (0.0) to the top (1.0).
vertical offset	count	> 0	0	Number of counts reported for vertical position when the joystick is at the bottom of travel.
vertical threshold	%	$\geq 0, \leq 1$	0.1	Vertical "deadspace" around joystick center.

Where the direction entry is `axis` *N*, the user is referring to a specific axis of the joystick. Direction `horizontal` is a synonym for `axis 0`, `vertical` for `axis 1`,

and any additional axes supported by the joystick can be referenced here.

The slope and offset values are used to map joystick position in counts to its displacement as a percentage of total travel, as shown in equation 2.

$$\text{displacement}_{\%} = \frac{\text{position}_{counts} - \text{offset}}{\text{slope}} \tag{2}$$

Although this equation is used to calibrate the joystick movement, the coordinate system ultimately used by the kinematics is slightly different. For the purposes of the joystick axis mappings in 3.3.5.1, the joystick input is a signed percentage of displacement from the center location measured against the total possible displacement in the given direction. So a joystick in the most lower left-hand corner would be in position (-1.0, -1.0), in the upper right-hand it would be (1.0, 1.0), and in the center (0.0, 0.0).

All joysticks produce a certain amount of random noise when they are centered. The "deadspace" value defines the width of the window around the center point wherein values are considered to be purely noise. Thus, any joystick displacement around 0.0 ± threshold will be considered to be 0.0, that is, centered.

3.1.5 XML Format for Equipment

Here is a template of the equipment configuration XML files produced by the `trackerxml` program and used as input to the tracker program. This file may also be created manually, but in that case, the tracker program does not check for inconsistent entries; errors in manually created or edited files may create unexpected performance.

```
<equipment>
    <screen   horizontalDistance="horizontal distance"
                verticalDistance="vertical distance"
           horizontalResolution="horizontal resolution"
             verticalResolution="vertical resolution"
                verticalRefresh="vertical refresh"/>
    <clock            resolution="clock resolution"/>
    <mouse>
        <horizontal  slope=horizontal slope"
                    offset=horizontal offset"/>
        <vertical     slope=vertical slope"
                    offset=vertical offset"/>
    </mouse>
```

```
    <joystick>
        <axis number="0"
                        slope=horizontal slope"
                       offset=horizontal offset"
                    threshold=horizontal threshold"/>
        <axis number="1"
                        slope=vertical slope"
                       offset=vertical offset"
                    threshold=vertical threshold"/>
        <axis number="N"
                        slope=slope for N"
                       offset=offset for N"
                    threshold=threshold for N"/>
    </joystick>>
</equipment>
```

3.2 Subject

The subject information is, from the view of the tracker program, an unformatted block of data. It may contain any number of lines, tabs or other characters. It will be imported directly into the output file without modification.

3.2.1 Text Format

Anything but embedded XML is not allowed. If desired, this can be added later.

3.2.2 XML Format for Subject

```
<subject>
    ...input text file contents...
</subject>
```

3.3 Trial

Trial parameters are those which differ from trial to trial and are used to describe the independent motion or disturbance of the cursor and target, how the user inputs are to be interpreted and the acceptable timing and error limits.

There are four categories of entries in this file:

1. general settings: setting the rates of execution and the termination criteria
2. graphics: setting the size of the screen, cursor and target and the colours

3. user inputs: defining input devices, gains and latencies

4. kinematics: the equations of motion for the target motion and cursor disturbance

3.3.1 General Settings

Entity	Function	Measure	Value
`rate`	`configuration`	`logging`	*`value`*
`rate`	`configuration`	`acquisition`	*`value`*
`termination`	`configuration`	`time`	*`limit`*
`delimiter`	`configuration`	*`value`*	

Parameter	Units	Range	Default	Description
logging rate	Hz	> 0	100	Rate at which data samples are saved.
acquisition rate	Hz	>logging rate	logging rate	Rate at which user inputs are sampled and kinematics calculated.
termination limit	seconds	> 0	none	Elapsed time by which the trial is to terminate.
delimiter	symbol	`tab` `comma`	tab	Field delimiter used by output file.

The experimenter can set the desired data logging rate and the data acquisition rate. The logging rate is the rate at which data is stored to disk, while the data acquisition rate is rate at which the input devices are sampled and the kinematics are recalculated.

The acquisition rate must be faster than the logging rate; if specified to be slower, the acquisition rate will be increased to match the logging rate. No averaging of sample values during an acquisition window between logging events is performed at this time.

The acquisition rate determines the time epsilon (the intersample interval) which should be the shortest period of that the experimenter cares about. If this value is larger than the inter-refresh interval on the monitor, the refresh interval will be used since the program needs to service every frame of the screen.

If no specific termination criterion is given, the trial will end when an input mapped to the `exit` function (3.3.5.1) is activated. If a time limit is given (in terms of seconds lapsed accurate to the intersample interval), then the trial will end when that period of time has passed or the `exit` function is activated, whichever comes first.

3.3.2 Graphics

We divide the graphics elements into two sections: configuration of the window and the configuration of the target and cursor.

3.3.3 Window Specification

Entity	Function	Measure	Value
window	configuration	horizontal	*value*
window	configuration	vertical	*value*
window	configuration	colour	*value*
window	configuration	fullscreen	*value*

Parameter	Units	Range	Default	Description
window horizontal	cm	> 0	screen width	Width of the window in which to draw the cursor and target.
window vertical	cm	> 0	screen height	Height of the window in which to draw the cursor and target.
window colour	defined colour, RGBA	symbolic constant, $0 \leq C \leq 1$	black	Background colour of the window.
window fullscreen	boolean	yes, no	yes	Indicates whether or not the full screen should be used or if the program should run within a window.

The user may use a predefined colour constant or specify four floating point values between 0.0 and 1.0 defining a new colour's red, green and blue components and alpha value. The red, green and blue values define the colour, while the alpha defines the intensity. The values must be in a single cell, separated by commas, with optional parentheses around the whole.

The predefined settings and their RGBA equivalents are:

Constant	R	G	B	A
white	1.0	1.0	1.0	1.0
black	0.0	0.0	0.0	0.0
red	1.0	0.0	0.0	1.0
orange	1.0	0.5	0.0	1.0
yellow	1.0	1.0	0.0	1.0
green	0.0	1.0	0.0	1.0
blue	0.0	0.0	1.0	1.0
purple	1.0	0.0	1.0	1.0

3.3.4 Target and Cursor Specification

Entity	Function	Measure	Value(s)				
`target`	`configuration`	`shape`	`chevron`	*`width`*	*`height`*	*`thickness`*	*`colour`*
`cursor`	`configuration`	`shape`	`chevron`	*`width`*	*`height`*	*`thickness`*	*`colour`*

Parameter	Units	Range	Default	Description
measure	predefined shape	`chevron`	*none*	Shape of the target or cursor to use followed by shape-specific parameters.
width	cm	≥ 0	*none*	Horizontal width of the chevron.
height	cm	≥ 0	*none*	Vertical height of the chevron.
thickness	cm	≥ 0	*none*	Vertical thickness of the chevron.
colour	defined colour, RGBA	symbolic constant, $0 \leq C \leq 1$	*none*	Colour of the chevron.

The experimenter can choose the shape, dimensions and colour of the target and cursor objects. Currently, the only implemented shape is `chevron`.

The dimensions are used to define the chevron as illustrated in figure 1.

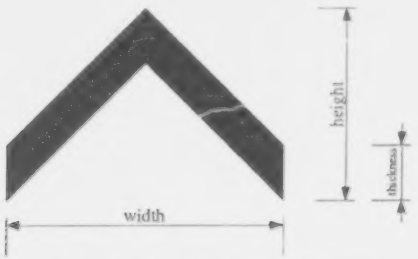


Figure 1: *Chevron dimensions*

To add a new shape to the tracker requires creating a C++ class for that shape, then modifying the rest of the program to handle the new class. The steps are:

1. Create a new drawing class which implements the `Drawable` abstract interface, using OpenGL commands to draw the shape.

2. Create a `configure` method within the new class that can obtain its configuration information from the XML trial configuration.

3. Modify `main` to instantiate the new shape and pass it the cursor or target XML configuration pointer.

Using the existing `Chevron` class as a template, adding a new shape should be quite straightforward.

3.3.5 User Input

We divide user input configuration into configuring the three input devices: joystick, mouse and keyboard.

Joystick axes are the simplest. By applying the calibration equations established in 3.1.4, we can obtain joystick position in terms of percentage of displacement from center. So, if the joystick is moved all the way to the upper-right corner, the joystick position is (1.0, 1.0), that is, 100% of right horizontal displacement, 100% of upward vertical displacement. Similarly, moving the joystick all the way to the lower-left corner would give a position of (-1.0, -1.0), the sign of the value indicating the direction of the displacement.

Mouse axes are measured in terms of movement of the system mouse cursor away from the center of the screen. The system mouse cursor is a cursor controlled by the operating system and typically visible as a small arrow, as distinct from the tracker cursor which is controlled by the tracker program according to the user input configuration. Although the system mouse cursor is not visible, it is still controlled by the subject's mouse in the usual manner. We measure the offset not in terms of percentage displacement as with the joystick, but in terms of centimetres of displacement, using the screen dimensions to calibrate this value. This displacement is signed in the same manner as the joystick, so moving the cursor to the right 5cm and down 2cm will give a mouse position of (5.0, -2.0). There is no limit to this displacement.

The experimenter can configure multiple input equations, each with their own set of parameters. The results of the equations are added together to produce the final mapping from user input to cursor movement.

Buttons, whether joystick buttons, mouse buttons or keyboard keys, are slightly more complex. To interpret these correctly, we define the concept of a single button press. A button press has occurred if the button or key is down when the acquisition loop cycles. Thus, if a specific key is held down for 50 cycles, it counts as 50 button presses. If the acquisition rate (3.3.1) is 100 Hz, then holding the button down for 0.5 seconds would give 50 button presses. The units of the button/key mapping given in 3.3.5.1, 3.3.5.2 and 3.3.5.3 are button presses.

For data logging purposes, when we given the button/key press count, it is *not* in terms of "button presses" as defined above but in actual physical presses.

3.3.5.1 Joystick

Entity	Function	Device Axis	Cursor Axis	Order	Slope	Offset	Delay
`joystick`	`map`	`horizontal (axis 0)`	*value*	*value*	*value*	*value*	*value*
`joystick`	`map`	`vertical (axis 1)`	*value*	*value*	*value*	*value*	*value*
`joystick`	`map`	`axis` *N*	*value*	*value*	*value*	*value*	*value*
`joystick`	`map`	`button` *N*	*value*	*value*	*value*	*value*	*value*

Parameter	Units	Range	Default	Description

device axis	configured axis	`axis` *N*
		`horizontal`
		`vertical`
			0	Defined joystick axis or button.

cursor axis	known axis	`x y r`
		`exit`
		`start`
			none	Axis of the cursor to which the input is to be mapped.

order	known order	`position`
		`velocity`
		`acceleration`
		$N = 0, 1, 2$
			`position`	Indicates whether the mapping applies to the cursor axis as a change in position (order = 0), velocity (order = 1) or acceleration (order = 2).

slope	$\frac{\text{U}}{\%}$ $\frac{\text{U/s}}{\%}$ $\frac{\text{U/s}^2}{\%}$	any	1	Multiplier of the displacement along the device axis.
offset	U U/s U/s^2	any	0	Linear offset applied to the displacement along the device axis.
delay	milliseconds	≥ 0	0	Amount of delay induced within input from this axis.

The induced delay will be accurate to the intersample interval defined by the acquisition rate in 3.3.1.

We map input devices axes or buttons onto the motion of the cursor. For input axes, the values are be linearly scaled using equation 3 where x represents the linear displacement as a percentage of total displacement along the device axis and y represents the value of appropriate order (that is, the current position, velocity or acceleration) along the cursor axis.

$$\text{output} = \frac{\text{input}_{\text{counts}} - \text{offset}}{\text{slope}} \tag{3}$$

The value of U in table 3.3.5.1 depends on the axis to which the mapping is applied. U is measured in centimetres when applied to `x` or `y`, in radians when applied to `r`. The units of the slope offset are a function of the order to which the mapping is applied, whether U, U/s or U/s^2.

If the input "axis" is a joystick button, then each acquisition cycle in which the button is depressed introduces a fixed changed of appropriate order in the mapped axis according to the user-defined offset as shown in equation 4, where y_t is the value of appropriate order along the cursor axis at the current time t and y_{t-1} is the same value at time $t - 1$.

$$y_t = y_{t-1} + \begin{cases} 0 & \text{not pressed} \\ b & \text{pressed} \end{cases} \tag{4}$$

3.3.5.2 Mouse

Entity	Function	Device Axis	Cursor Axis	Order	Slope	Offset	Delay
`mouse`	`map`	`horizontal`	*value*	*value*	*value*	*value*	*value*
`mouse`	`map`	`vertical`	*value*	*value*	*value*	*value*	*value*
`mouse`	`map`	`button` *N*	*value*	*value*	*value*	*value*	*value*

Parameter	Units	Range	Default	Description

device axis	configured axis	`horizontal`
		`vertical`
		`button` *N*
			none	Defined mouse axis or button.

cursor axis	known axis	`x y r`
		`exit`
		`start`
			none	Axis of the cursor to which the input is to be mapped.

order	known order	`position`
		`velocity`
		`acceleration`
		$N = 0, 1, 2$
			`position`	Indicates whether the mapping applies to the cursor axis as a change in position (0), velocity (1) or acceleration (2).

slope	$\frac{\text{U}}{\%}$ $\frac{\text{U/s}}{\%}$ $\frac{\text{U/s}^2}{\%}$	any	1	Multiplier of the displacement along the device axis.
offset	U U/s U/s^2	any	0	Linear offset applied to the displacement along the device axis.
delay	milliseconds	≥ 0	0	Amount of delay induced within input from this axis.

With the mouse, the input value represents a physical displacement in centimetres from its initial position. The value of U is the same as for the joystick (3.3.5.1). Note that with the mouse, the positive direction of displacement is downward while the negative is upward.

3.3.5.3 Keyboard

Entity	Function	Device Axis	Cursor Axis	Order	Slope	Offset	Delay
`keyboard`	`map`	*`symbol`*	*`value`*	*`value`*	`none`	*`value`*	*`value`*

Parameter	Units	Range	Default	Description

device axis	known key	`a ... z`
		`0 ... 9`
		`left`
		`right`
		`escape`
		`space`
			none	Keyboard keys including normal letters, left and right arrow keys, escape and spacebar.

cursor axis	known axis	`x y r`
		`exit`
		`start`
			none	Axis of the cursor to which the input is to be mapped.

order	known order	`position`
		`velocity`
		`acceleration`
		$N = 0, 1, 2$
			`position`	Indicates whether the mapping applies to the cursor axis as a change in position (0), velocity (1) or acceleration (2).

slope	n/a	n/a	n/a	Linear value of the appropriate order applied to the cursor axis.
offset	U U/s U/s^2	any	0	Does not apply with keys.
delay	milliseconds	≥ 0	0	Amount of delay induced within input from this axis.

The input from each key, if not mapped to a function axis, is interpreted in the same manner as the joystick buttons (3.3.5.1) and mouse buttons using equation 4. The value of U is interpreted similarly.

If the cursor axis is either `start` or `exit`, then these "axes" are functions performed by the system when the key is pressed. Function `exit` will cause the trial to end, while function `start` is used to start the trial once the system is initialized. For function axes, the values for slope, offset and order are not applicable. Delay is still applied.

3.3.6 Kinematics

We divide kinematics into three sections: (i) movement and mass of axes, (ii) motion epsilons and (iii) motion equations.

3.3.6.1 Movement and Mass of Axes

Entity	Function	Axis	Free	Mass
`target`	`freedom`	*`value`*	*`value`*	*`value`*
`cursor`	`freedom`	*`value`*	*`value`*	*`value`*

Parameter	Units	Range	Default	Description
axis	known axis	`x y r`	*none*	Axis of the target or cursor that we are describing.
free	boolean	`yes` `no`	none	Defines whether the target or cursor is free to move on this axis.
mass	none	≥ 0	none	Scaling factor applied to the acceleration which acts as a mass for acceleration. Does not apply if there is no freedom of motion on this axis.

3.3.6.2 Motion Epsilons

Entity	Function	Axis	Order	Epsilon
`target`	`epsilon`	*value*	*value*	*value*
`cursor`	`epsilon`	*value*	*value*	*value*

Parameter	Units	Range	Default	Description
axis	known axis	`x y r`	*none*	Axis of the target or cursor that we are describing.
order	known order	`position` `velocity` `acceleration` $N = 0, 1, 2$	none	Indicates whether the epsilon applies to the axis in terms of position (0), velocity (1) or acceleration (2).
epsilon	U U/s U/s^2	≥ 0	various	Defines the minimum magnitude of the motion of the appropriate order.

An *epsilon value* is the limit limit at which, if a number is smaller in magnitude than a defined epsilon, that number is considered to be zero. They are used within tracker to suppress tiny artifact motions. For example, the experimenter may wish to set rotation velocity epsilon to 0.01 radians per second. This means that if the cursor rotation velocity, due say to user input, drops below 0.01 rads/sec, the cursor will stop rotating, rather than continue to rotate very slowly.

The tracker program will generate defaults for these values if they are not established explicitly. Typically there is no reason to explicitly define position epsilons, since the limited resolution of the display device implicitly does that (you can't move less than a pixel in position). Velocity epsilons are generated so that if the velocity of cursor or target is such that there is no change in position between successive frames, the velocity is considered 0. Acceleration epsilons are chosen so if there is no change in position within 0.1 seconds, acceleration is considered zero.

3.3.6.3 Motion Equations

Motion equations are used to describe the movement of the target and the disturbance of the cursor. As with the user input equations, the experimenter can provide multiple equations, each with their own set of parameters. The results of each equation are added together to produce the final motion or disturbance of the target and cursor.

Entity	Function	Axis	Order	Equation	p0	p1	p2	p3
`target`	`motion`	*value*	*value*	*value*	*value*	*value*	*value*	*value*
`cursor`	`motion`	*value*	*value*	*value*	*value*	*value*	*value*	*value*

Parameter	Units	Range	Default	Description
axis	known axis	`x y r`	*none*	Axis of the cursor to which the sine is to be mapped.

order	known order	`position`
		`velocity`
		`acceleration`
		$N = 0, 1, 2$
			`position`	Indicates whether the mapping applies to the axis as a change in position (0), velocity (1) or acceleration (2).

equation	known equation	`sine` `step`	none	Type of equation to be used.
p0 ... p3	various	various	various	Parameters to the selected equation. The number and usage of the parameters depends on the equation type.

The two equations currently supported are the continuous `sine` equation and the discrete `step` equation.

The `sine` equation accepts four parameters:

Parameter	Name	Units	Range	Description
p0	b	U U/s U/s2	any	Scalar offset added at the appropriate order.
p1	m	none	any	Weighting value for the sinusoid.
p2	ω	seconds^{-1}	any	Controls the period of the sinusoid.
p3	ϕ	none	any	Controls the phase of the sinusoid.

Together, these parameters give us equation 5 where y = motion value added at the appropriate order to the selected entity and t is the elapsed time in milliseconds.

$$y = b + m\sin(\omega t + \phi) \tag{5}$$

The `step` equation accepts four parameters:

Parameter	Name	Units	Range	Description
p0	amplitude	U U/s U/s^2	any	Scalar offset added at the appropriate order.
p1	average time	milliseconds	> 0	Average elapsed time at which the step should be implemented.
p2	time window	milliseconds	≥ 0	Acceptable variation in time of the random presentation of the step.
p3	step label	any	any	Label assigned to this step. Saved in the data file when the step is implemented.

When configured, the system will choose a random trigger time t within the range of p1 $\pm$ p2. When activated, the scalar offset p0 will be added at the appropriate order (typically position) on the selected axis of the selected entity, when the sign of the offset being adjusted so as to keep the entity within the screen limits.

3.3.7 XML Format for Trial

```
<trial>
    <rate type="logging"     value="logging rate"/>
    <rate type="acquisition" value="acquisition rate"/>

    <window horizontalSize="window horizontal size"
              verticalSize="window vertical size"
                 colourRed="window background red component"
               colourGreen="window background green component"
                colourBlue="window background blue component"
               colourAlpha="window background alpha component"
                fullScreen="no" />

    <termination type="time"
                 limit="time limit"/>
    <delimiter symbol="tab"/>

    <target>
        <shape type="shape type"
             height="target height"
              width="target width"
          thickness="target thickness"
          colourRed="target colour red component"
        colourGreen="target colour green component"
         colourBlue="target colour blue component"
        colourAlpha="target colour alpha component"/>

        <axis name="axis name"
```

```
            free="axis free flag"
            mass="axis mass">
          <epsilon order="applied order"
                   value="epsilon value"/>
    </axis>

    <equation  axis="applied axis"
              order="applied order">
        <sine offset="offset value"
              weight="weight value"
              period="period value"
               phase="phase value" />

        <step amplitude="amplitude value"
                    time="average time value"
                  window="time window width"
                   label="label of step" />
    </equation>

</target>

<cursor>
    <shape type="shape type"
         height="target height"
          width="target width"
      thickness="target thickness"
      colourRed="target colour red component"
    colourGreen="target colour green component"
     colourBlue="target colour blue component"
    colourAlpha="target colour alpha component"/>

    <axis name="axis name"
          free="axis free flag"
          mass="axis mass"/>
    <equation  axis="applied axis"
              order="applied order">
        <sine offset="offset value"
              weight="weight value"
              period="period value"
               phase="phase value" />
    </equation>

    <equation axis ="applied axis"
              order="applied order">
        <step amplitude="amplitude value"
```

```
                time="average time value"
               window="time window width"
                label="label of step" />
        </equation>

        <joystick>
            <axis       name="device axis"
                  cursorAxis="cursor axis"
                       order="order value"
                       delay="delay value">
                <linear offset="offset"
                         slope="slope"/>
                <statistic type="keypress"/>
                <statistic type="axis"/>
            </axis>
        </joystick>
        <mouse>
            <axis       name="device axis"
                  cursorAxis="cursor axis"
                       order="order value"
                       delay="delay value">
                <linear offset="offset"
                         slope="slope"/>
                <statistic type="keypress"/>
                <statistic type="axis"/>
            </axis>
        </mouse>
        <keyboard>
            <axis       name="device axis"
                  cursorAxis="cursor axis"
                       order="order value"
                       delay="delay value">
                <linear offset="offset"/>
                         slope="slope"/>
                <statistic type="keypress"/>
            </axis>
        </mouse>
    </cursor>

</trial>
```

Annex A
Example Files

A.1 Equipment Configuration

"Dell Inspiron 8100 Test System"					
clock	calibration	resolution	10		
"15 XGA LCD Screen"					
screen	calibration	horizontal	distance	30.5	
screen	calibration	vertical	distance	22.9	
screen	calibration	horizontal	resolution	1600	
screen	calibration	vertical	resolution	1400	
screen	calibration	vertical	refresh	85	
"Logitech PS/2 Mouse"					
mouse	calibration	horizontal	85	0	
mouse	calibration	vertical	82	0	
"MS FlightMaster Gameport Joystick"					
joystick	calibration	horizontal	32122	32122	0.1
joystick	calibration	vertical	32123	32123	0.1

A.2 Subject Configuration

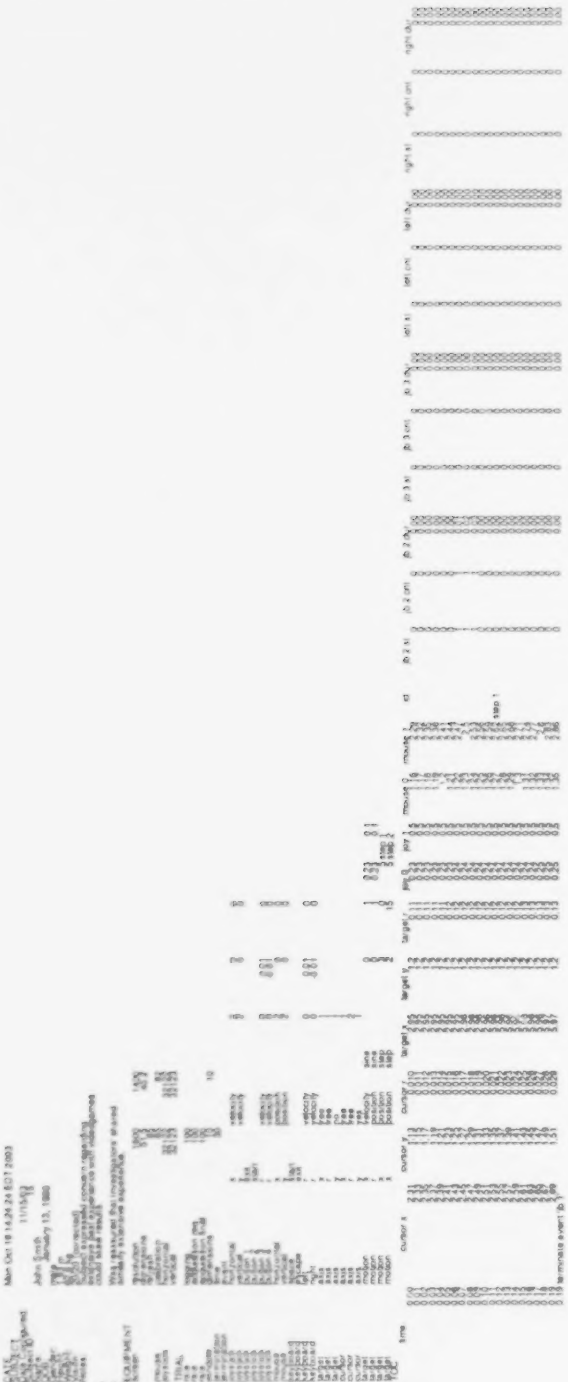
"Subject #15 Configuration"	
Date Configured:	November 15, 2003
Subject ID:	15
Name:	John Smith
DOB:	January 13, 1960
Gender:	male
Height:	1.93 m
Weight:	62.6 kg
Vision:	20/20 (corrected)
Notes:	Subject expressed concern regarding extensive past experience with videogames could skew the results.
	Was reassured that investigators' shared similarly extensive experience.

A.3 Trial Configuration

"Trial 1825 Configuration"								
"General Settings"								
"entity"	"function"	"measure"	"value"					
rate	configuration	logging	100					
rate	configuration	acquisition	100					
window	configuration	horizontal	10					
window	configuration	vertical	10					
window	configuration	colour	(0.32, 0.12, 0.23, 0.23)					
termination	configuration	time	30					
"Joystick Controls Velocity and Rotation"								
"entity"	"function"	"from axis"	"to axis"	"order"	"slope"	"offset"	"delay"	
joystick	map	horizontal	x	velocity	2.0	0.0	0	
joystick	map	vertical	y	velocity	2.0	0.0	0	
joystick	map	button 0	exit					
joystick	map	button 1	start					
joystick	map	button 2	r	velocity	0	0.01	0	
joystick	map	button 3	r	velocity	0	-0.01	0	
"Mouse Controls Position"								
"entity"	"function"	"from axis"	"to axis"	"order"	"slope"	"offset"	"delay"	
mouse	map	horizontal	x	position	2.0	0.0	0	
mouse	map	vertical	y	position	2.0	0.0	0	
"Keyboard Controls Rotation"								
"entity"	"function"	"from axis"	"to axis"	"order"	"slope"	"offset"	"delay"	
keyboard	map	space	start					
keyboard	map	escape	exit					
keyboard	map	left	r	velocity	0	0.01	0	
keyboard	map	right	r	velocity	0	-0.01	0	
"Target Moves Horizontally and Rotates"								
"entity"	"function"	"measure"	"free?"	"mass"				
target	freedom	x	yes	1.0				
target	freedom	r	yes	1.0				
target	freedom	y	no	1.0				
"Target Characteristics"								
"entity"	"function"	"measure"	"order"	"value"				
target	epsilon	x	1	0.01				
target	epsilon	y	1	0.01				
target	epsilon	z	1	0.001				
"entity"	"function"	"measure"	"value"					
target	configuration	colour	white					
target	configuration	height	3.0					
target	configuration	width	4.0					

"Cursor Has Full Freedom"								
"entity"	"function"	"measure"	"free?"	"mass"				
cursor	freedom	x	yes	1.0				
cursor	freedom	r	yes	2.0				
cursor	freedom	y	yes	1.0				
"Cursor Characteristics"								
"entity"	"function"	"measure"	"order"	"value"				
cursor	epsilon	x	1	0.01				
cursor	epsilon	y	1	0.01				
cursor	epsilon	z	1	0.001				
"entity"	"function"	"measure"	"value"					
cursor	configuration	colour	yellow					
cursor	configuration	height	3.0					
cursor	configuration	width	4.0					
"Target Motion"								
"entity"	"function"	"axis"	"order"	"eqn"	"offset"	"weight"	"period"	"ph
target	motion	x	velocity	sine	0.0	1.0	0.23	0.1
target	motion	r	position	sine	0.0	1.0	0.23	0.1
"entity"	"function"	"axis"	"order"	"eqn"	"amp"	"time"	"width"	"la
target	motion	x	position	step	3.0	0.0	5.0	ste
target	motion	x	position	step	2.0	15.0	5.0	ste

A.4 Output Data File



Annex B
Recommended Platform

The following hardware was found sufficient to meet the timing and I/O requirements under Linux RedHat 8.0. This system should also be sufficient for using tracker under Windows.

CPU:	Pentium IIIm, 996 MHz
Video:	nVidia mobile Chipset
Hard disk:	5700 RPM IDE
Joystick:	Microsoft Sidewinder Force Feedback2 (USB)
Mouse:	Logitech MouseMan+ (USB)

Annex C
Software Requirements and Procedures

The RPM provided on the tracker distribution CD was built on Fedora Linux. This package should also install on RedHat 9. Packages for other version of RedHat Linux were not provided, since these versions are no longer supported by RedHat Inc. The distribution CD includes the source RPM for the tracker program, and the two required third party RPM packages, and their source tarballs, thus building the tracker program on any past or future version of Linux (within reason) is possible. Thus it is possible to build compatible RPM packages for those versions.

On an RPM based Linux platform such as Fedora, the required dependencies are best satisfied by downloading compatible RPM packages from the distribution CD, in the case of OS provided packages, from the tracker distribution CD, from the author, or an RPM repository such as www.rpmfind.net. Embedded within the RPM package is the list of dependencies, so any missing dependencies will be identified at installation.

The following OS packages are required to run the tracker program:

- `glibc`,
- `libgcc`,
- `libstdc++`,
- `SDL`, and
- `XFree86-Mesa-libGL` for the OpenGL libraries.

The packages `glibc` and `libgcc` are part of the default installation of Fedora, while libstdc++, SDL and XFree86-Mesa-libGL may be installed depending on the package options specified at the time of installation.

With the required dependencies installed, the tracker RPM package will install with the normal RPM command.

To build a binary tracker RPM package from the source RPM package, i the following additional OS RPM packages are required:

- `gcc`,
- `glibc-devel`,
- `libstdc++-devel`,
- `SDL-devel`, and
- `XFree86-devel` for the OpenGL library and header files,

and the following third party RPM packages are required:

- `mxml`, (versions 1.1 or newer) and
- `nana`, (version 2.5).

These packages have been provided on the tracker distribution CD. There is one caveat with these third party RPMs, and that is that the packager may assume that a particular version of dependent libraries are installed on the target system. For example, the RPM for mxml that is downloaded from the developers website is built against glibc version 2.3, while RedHat 7.2 and 7.3, for example, are built against glibc version 2.2. In such a case, a binary RPM package for mxml needs to be built for the target OS. This can be done by obtaining the mxml source tarball, and running the following command:

```
> rpmbuild --tb mxml-1.3.tar.gz
```

Once these dependent packages have been installed, the following command will build the binary rpm:

```
> rpmbuild --rebuild tracker-1.0.0-1.src.rpm
```

For unsupported versions of RedHat, in particular RedHat 7.2 and RedHat 7.3, the dependency on the OpenGL shared library is problematic, because the package containing this library changed from RedHat 7.2 and 7.3, and again between RedHat 7.3 and RedHat 9/Fedora. This can be addressed by modifying the .spec file contained within the source RPM and either remove the dependency on XFree86-Mesa-libGL, or replace it with Mesa for RedHat 7.2, or XFree86-libs for RedHat 7.3.

On an RPM based Linux OS, source RPMS automate the process of building programs for installation. An alternative is to manually build from source. Programs using the autoconf/automake facility (which most do) can be compiled and installed by expanding the tarball, then in the top level directory, running the following sequence of commands:

```
> ./configure
> make
> make install
```

This procedure is not required, moreover not recommended for RPM packages available from the OS distribution itself, but is an option for the third party packages nana and mxml, and the tracker program itself. Packages built this way will typically install programs under /usr/local/bin, and libraries under /usr/local/lib, so these paths may need to be explicitly provided on the `./configure` command line if they cannot be automatically found by that script.

The following software components were used with Windows:

- SDL-1.2.6
- OpenGL32 provided as part of the standard Windows installation
- Cygwin-1.5.6.1
- mxml-1.1.1
- nana-2.5

Because Cygwin is a compatibility library, not an RPM based Linux distribution, and certain libraries are not part of a Cygwin installation, the tracker program will have to be built manually. This will be required for mxml, nana, SDL and the tracker program. The procedure for building each is the same as for a manual compile/install under GNU/Linux, namely:

```
> tar xzvf packagename-version.tar.gz
> cd packagename-version
> ./configure
> make
> make install
```

References

1. (2003). Open Graphics Library OpenGL (Online). dunno. http://www.opengl.org (October 1, 2003).
2. (2003). Simple MediaDirect Layer (SDL) (Online). dunno. http://www.libsdl.org (October 1, 2003).
3. (2003). Mini-XML Home Page (Online). Michael Sweet. http://www.easysw.com/ mike/mxml (October 30, 2003).
4. (2000). Extensible Markup Language (XML) 1.0 (Second Edition) (Online). W3C World Wide Web Consortium. http://www.w3.org/TR/REC-xml (June 17, 2003).

Index

axis
  joystick, 19
  mass, 22
  mouse, 20

button
  coordinates, 17

C++, 1
chevron, 16
clock
  resolution, 10
colour settings, 15
configuration
  equipment, 1, 4
  subject, 1, 7, 13
  trial, 1, 8, 13
cursor
  colour, 16
  dimensions, 16
  shape, 16

degrees of freedom, 22
delay, 18
delimiter, 14

equation
  sine, 23
  step, 23, 24

functions, 21

intersample interval, 14

joystick
  calibration, 6, 11
  coordinates, 12, 17
  digital, 6
  kinematic mapping, 18
  threshold, 11

keyboard
  kinematic mappings, 21
kinematic equations, 23

Linux, 1

mass, 22
mouse
  calibration, 6, 11
  coordinates, 17, 20
  kinematic mappings, 20
multiple trials, 9

OpenGL, 1
order, 18
output
  output.csv, 8
  tracker.log, 8
overwrite, 8

rate
  data acquisition, 14
  logging, 14

screen
  dimensions, 7, 10
  refresh rate, 10
  resolution, 10
SDL, 1
spreadsheet, 8

target
  colour, 16
  dimensions, 16
  shape, 16
termination, 14
timer
  resolution, 5
trackerxml, 1

video
  refresh rate, 5
  resolution, 5

window
  colour, 15
  dimensions, 15
Windows, 1

XML, 1

DOCUMENT CONTROL DATA SHEET

1a. PERFORMING AGENCY
Serenica Technologies Corporation, Toronto, ON

2. SECURITY CLASSIFICATION
UNCLASSIFIED
-

1b. PUBLISHING AGENCY
DRDC Toronto

3. TITLE
Tracker Program - User Guide

4. AUTHORS
Ed Kozak Tom Gee

5. DATE OF PUBLICATION
March 31 , 2004

6. NO. OF PAGES
44

7. DESCRIPTIVE NOTES

8. SPONSORING/MONITORING/CONTRACTING/TASKING AGENCY
Sponsoring Agency:
Monitoring Agency:
Contracting Agency : DRDC Toronto
Tasking Agency:

9. ORIGINATORS DOCUMENT NO.
Contract Report CR 2004-027

10. CONTRACT GRANT AND/OR PROJECT NO.
W7711-037866/001/TOR

11. OTHER DOCUMENT NOS.

12. DOCUMENT RELEASABILITY
Unlimited distribution

13. DOCUMENT ANNOUNCEMENT
Unlimited announcement

14. ABSTRACT

(U) This document is a user guide for and description of a computer-based, manual tracking task. The tracking task was developed to support experiments and modelling of human performance at DRDC Toronto. To this end, the program allows greater data sampling and collection rates than typically reported by other investigations within the literature; the acquisition and recording rates will depend upon the computer hardware and software used to run the tracking task program. The tracking task program is designed for use on Intel-based computers running Linux Redhat Fedora as well as Microsoft Windows NT operating systems. In order to work on Windows environments, additional, freely-available but non-standard software must be added such as Cygwin, mxml and SDL to execute; few modifications should be required for use on other Linux operating systems. The program provides pursuit, compensatory and general (pursuit and compensatory) tracking tasks. The experimenter can create a forcing function from a linear combination of any number of sinusoids or step inputs.

(U) Le présent document est le guide d'utilisateur et la description d'une tâche de poursuite manuelle informatisée. Cette tâche de poursuite a été mise au point pour soutenir les expériences et la modélisation de la performance humaine à RDDC Toronto. À cette fin, le programme permet de plus grandes fréquences d'échantillonnage et de collecte de données que celles qu'on observe généralement dans la documentation d'autres investigations; les taux d'acquisition et d'enregistrement dépendront du matériel et du logiciel utilisés pour exécuter le programme de tâches de poursuite. Le programme de tâches de poursuite est conçu pour des ordinateurs Intel tournant sous les systèmes d'exploitation Linux Redhat Fedora ainsi que Microsoft Windows NT. Afin de l'exécuter dans des environnements Windows, il faut ajouter des logiciels supplémentaires, gratuits mais non standard, comme Cygwin, mxml et SDL; peu de modifications devraient être nécessaires pour une utilisation sous d'autres systèmes d'exploitation Linux. Le programme offre des tâches de poursuite mobiles, compensatrices et générales (mobiles et compensatrices). L'expérimentateur peut créer une fonction de forçage à partir d'une combinaison linéaire de n'importe quel nombre de sinusoïdes ou de signaux d'entrée progressifs.

15. KEYWORDS, DESCRIPTORS or IDENTIFIERS

(U) manual tracking; experimentation; human performance